ITA 0443 - STATISTICS WITH R PROGRAMMING FOR REAL TIME PROBLEM

LAB RECORD

NAME:BARATH.P

REG NO :192121147

**Set-I**

**1. .(i) Write a function called kelvin\_to\_celsius() that takes a temperature in Kelvin and returns that**

**temperature in Celsius (Hint: To convert from Kelvin to Celsius you subtract 273.15)**

**(ii) Write suitable R code to compute the mean, median ,mode of the following values**

**c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)**

**(iii) Write R code to find 2nd**

**highest and 3rd Lowest value of above problem.**

**INPUT:**

**kelvin\_to\_celsius <- function(temp\_in\_k) {**

**temp\_in\_c <- temp\_in\_k - 273.15**

**return(temp\_in\_c)**

**}**

**x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)**

**# Mean**

**mean(x)**

**# Median**

**median(x)**

**# Mode**

**mode(x)**

**x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)**

**# 2nd highest**

**sort(x, decreasing = TRUE)[2]**

**# 3rd lowest**

**sort(x)[3]**

**OUTPUT:**

**> kelvin\_to\_celsius <- function(temp\_in\_k) {**

**+ temp\_in\_c <- temp\_in\_k - 273.15**

**+ return(temp\_in\_c)**

**+ }**

**>**

**> x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)**

**>**

**> # Mean**

**> mean(x)**

**[1] 60**

**>**

**> # Median**

**> median(x)**

**[1] 70**

**>**

**> # Mode**

**> mode(x)**

**[1] "numeric"**

**> x <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)**

**>**

**> # 2nd highest**

**> sort(x, decreasing = TRUE)[2]**

**[1] 90**

**>**

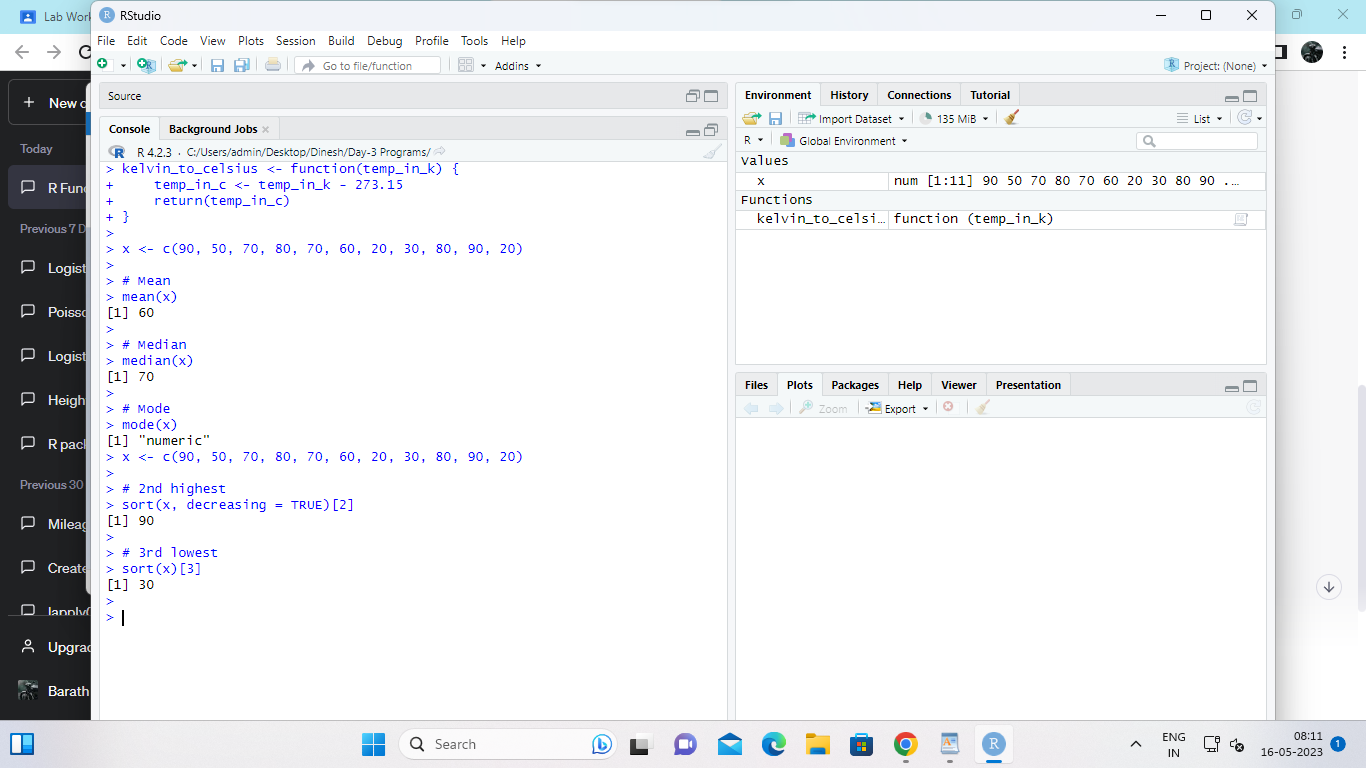
**> # 3rd lowest**

**> sort(x)[3]**

**[1] 30**

**>**

**>**

****

**2. Explore the airquality dataset. It contains daily air quality measurements from New York during a period**

**of five months:**

**• Ozone: mean ozone concentration (ppb),**

**• Solar.R: solar radiation (Langley),**

**• Wind: average wind speed (mph),**

**• Temp: maximum daily temperature in degrees Fahrenheit,**

**• Month: numeric month (May=5, June=6, and so on),**

**• Day: numeric day of the month (1-31).**

**i. Compute the mean temperature(don’t use build in function)**

**ii. Extract the first five rows from airquality.**

**iii. Extract all columns from airquality except Temp and Wind**

**iv. Which was the coldest day during the period?**

**v. How many days was the wind speed greater than 17 mph?**

**INPUT:**

**# Load the airquality dataset**

**data(airquality)**

**# (i) Compute the mean temperature without using the built-in function**

**mean\_temp <- sum(airquality$Temp)/nrow(airquality)**

**mean\_temp**

**# (ii) Extract the first five rows from airquality**

**head(airquality, 5)**

**# (iii) Extract all columns from airquality except Temp and Wind**

**airquality\_no\_temp\_wind <- airquality[, -c(4, 3)]**

**head(airquality\_no\_temp\_wind)**

**# (iv) Find the coldest day during the period**

**coldest\_day <- airquality$Day[which.min(airquality$Temp)]**

**coldest\_day**

**# (v) Count the number of days with wind speed greater than 17 mph**

**num\_windy\_days <- sum(airquality$Wind > 17)**

**num\_windy\_days**

**OUTPUT:**

**> # Load the airquality dataset**

**> data(airquality)**

**>**

**> # (i) Compute the mean temperature without using the built-in function**

**> mean\_temp <- sum(airquality$Temp)/nrow(airquality)**

**> mean\_temp**

**[1] 77.88235**

**>**

**> # (ii) Extract the first five rows from airquality**

**> head(airquality, 5)**

**Ozone Solar.R Wind Temp Month Day**

**1 41 190 7.4 67 5 1**

**2 36 118 8.0 72 5 2**

**3 12 149 12.6 74 5 3**

**4 18 313 11.5 62 5 4**

**5 NA NA 14.3 56 5 5**

**>**

**> # (iii) Extract all columns from airquality except Temp and Wind**

**> airquality\_no\_temp\_wind <- airquality[, -c(4, 3)]**

**> head(airquality\_no\_temp\_wind)**

**Ozone Solar.R Month Day**

**1 41 190 5 1**

**2 36 118 5 2**

**3 12 149 5 3**

**4 18 313 5 4**

**5 NA NA 5 5**

**6 28 NA 5 6**

**>**

**> # (iv) Find the coldest day during the period**

**> coldest\_day <- airquality$Day[which.min(airquality$Temp)]**

**> coldest\_day**

**[1] 5**

**>**

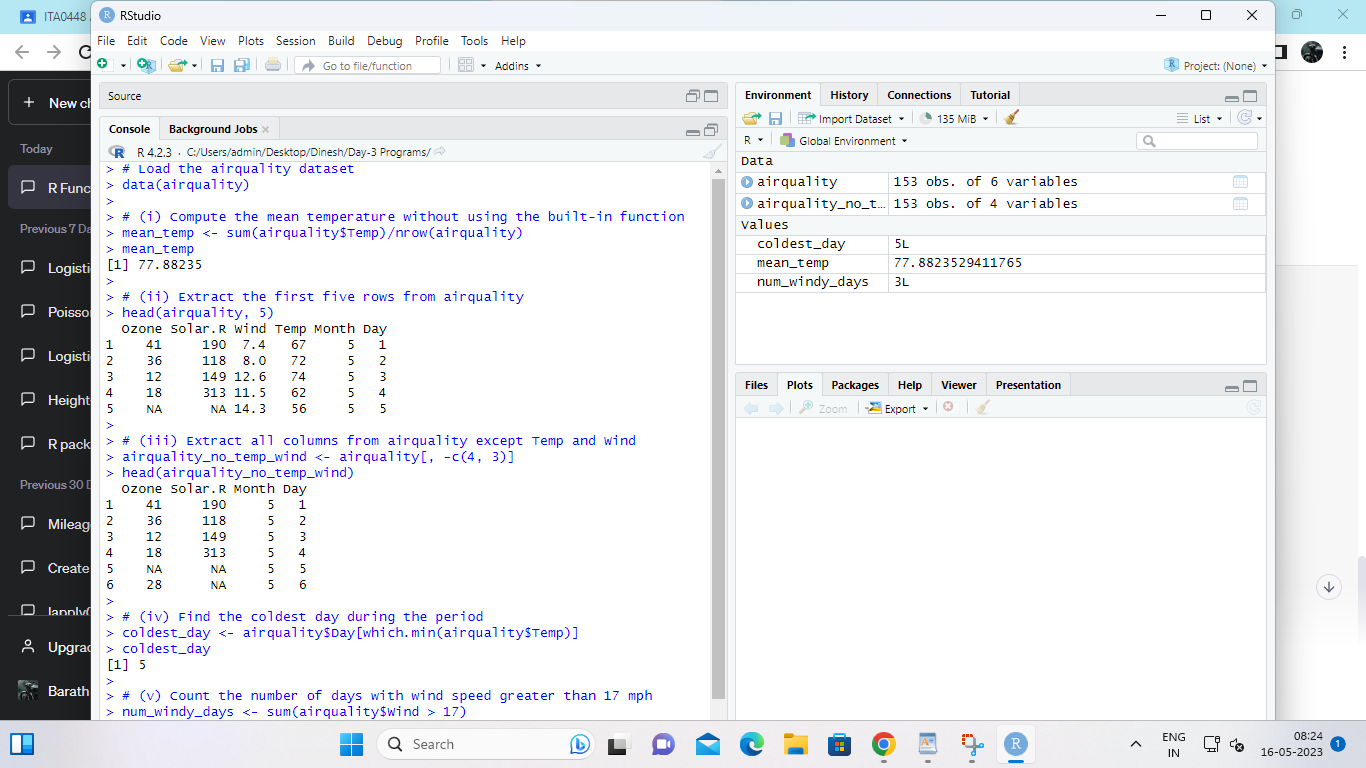
**> # (v) Count the number of days with wind speed greater than 17 mph**

**> num\_windy\_days <- sum(airquality$Wind > 17)**

**> num\_windy\_days**

**[1] 3**

**>**

****

3. (i) Get the Summary Statistics of air quality dataset

(ii)Melt airquality data set and display as a long – format data?

(iii)Melt airquality data and specify month and day to be “ID variables”?

(iv)Cast the molten airquality data set with respect to month and date features

(v) Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and

temperature per month?

INPUT;

summary(airquality)

library(reshape2)

airquality\_melted <- melt(airquality)

head(airquality\_melted)

airquality\_melted <- melt(airquality, id.vars = c("Month", "Day"))

head(airquality\_melted)

airquality\_casted <- dcast(airquality\_melted, Month + Day ~ variable)

head(airquality\_casted)

airquality\_avg <- dcast(airquality\_melted, Month ~ variable, mean)

airquality\_avg

OUTPUT;

> summary(airquality)

Ozone Solar.R Wind Temp

Min. : 1.00 Min. : 7.0 Min. : 1.700 Min. :56.00

1st Qu.: 18.00 1st Qu.:115.8 1st Qu.: 7.400 1st Qu.:72.00

Median : 31.50 Median :205.0 Median : 9.700 Median :79.00

Mean : 42.13 Mean :185.9 Mean : 9.958 Mean :77.88

3rd Qu.: 63.25 3rd Qu.:258.8 3rd Qu.:11.500 3rd Qu.:85.00

Max. :168.00 Max. :334.0 Max. :20.700 Max. :97.00

NA's :37 NA's :7

Month Day

Min. :5.000 Min. : 1.0

1st Qu.:6.000 1st Qu.: 8.0

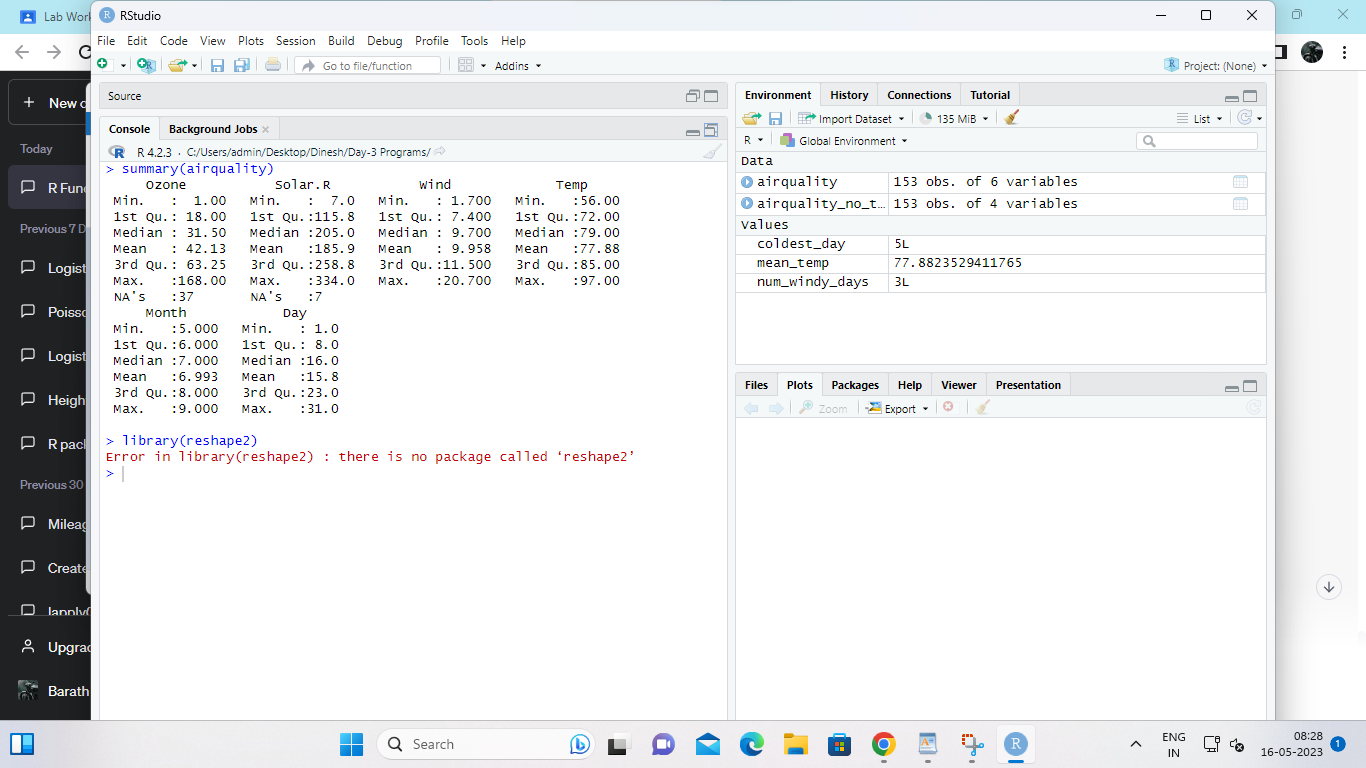
Median :7.000 Median :16.0

Mean :6.993 Mean :15.8

3rd Qu.:8.000 3rd Qu.:23.0

Max. :9.000 Max. :31.0

> library(reshape2)



4.(i) Find any missing values(na) in features and drop the missing values if its less than 10% else

replace that with mean of that feature.

(ii) Apply a linear regression algorithm using Least Squares Method on “Ozone” and “Solar.R”

(iii)Plot Scatter plot between Ozone and Solar and add regression line created by above model

INPUT:

# Read the data into a data frame

df <- read.csv("your\_file.csv")

# Check for missing values in each column

colSums(is.na(df))

# Replace missing values with mean if they are less than 10%

df[is.na(df) & colMeans(is.na(df)) < 0.1] <- sapply(df, mean, na.rm = TRUE)

# Drop missing values if they are more than 10%

df <- na.omit(df)

# Fit the linear regression model using the Least Squares Method

fit <- lm(Ozone ~ Solar.R, data = df)

# Print the summary of the model

summary(fit)

# Plot the scatter plot and add the regression line

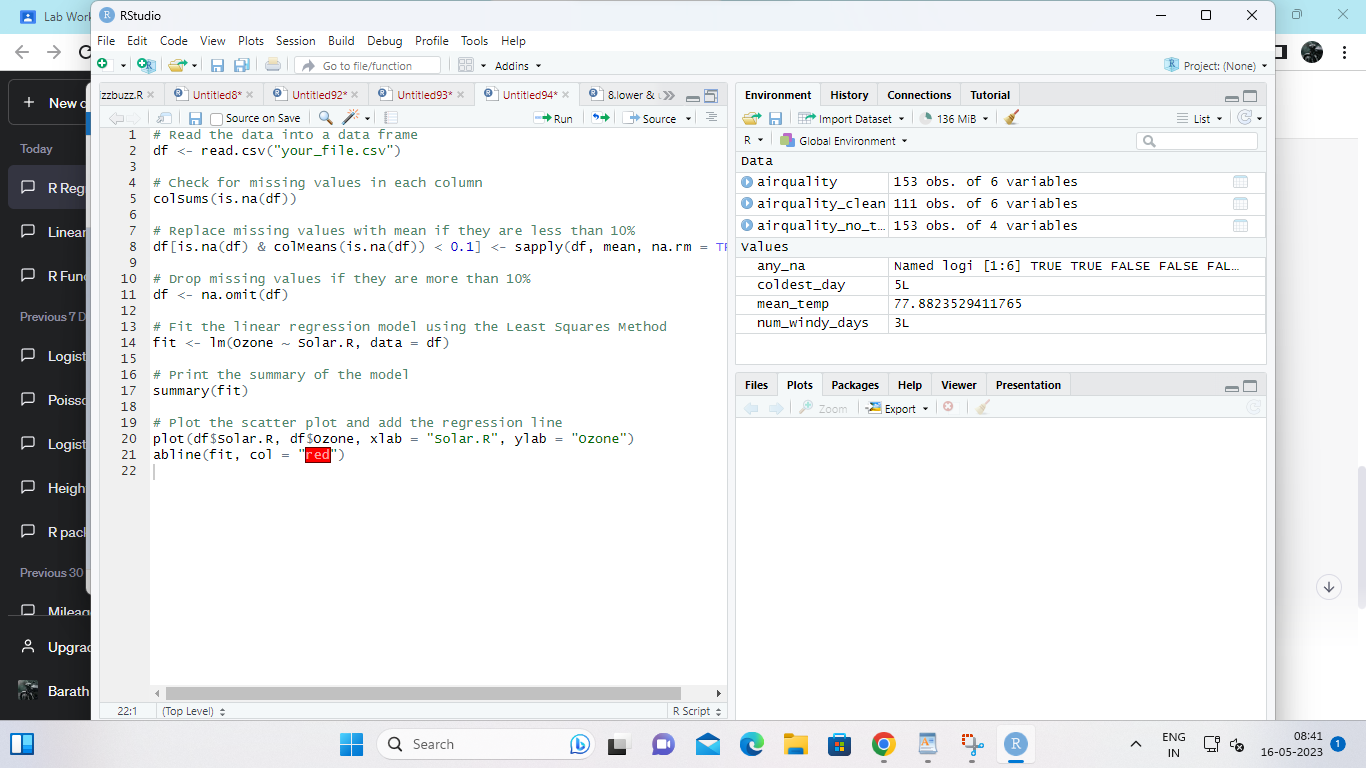
plot(df$Solar.R, df$Ozone, xlab = "Solar.R", ylab = "Ozone")

abline(fit, col = "red")

OUTPUT:

> # Read the data into a data frame

> df <- read.csv("your\_file.csv")



Set-II

1. (i)Write a function to find the factorial of a given number using “for” Loop

(ii) Create a 3x4 matrix with 12 random numbers between 1-100; have the matrix be filled our row-

by-row, instead of column-by-column. Name the columns of the matrix uno, dos, tres, cuatro, and

the rows x, y, z. Scale the matrix by 10 and save the result.

(iii) Extract the column called “uno” as a vector from the original matrix and save the result

INPUT:

# Define a function to find the factorial of a given number

factorial <- function(n) {

fact <- 1

for (i in 1:n) {

fact <- fact \* i

}

return(fact)

}

# Call the function with a sample value

factorial(5) # Output: 120

# Create a 3x4 matrix with 12 random numbers between 1-100, fill the matrix row-by-row, and name the columns and rows

set.seed(123) # Set the seed for reproducibility

matrix\_data <- matrix(sample(1:100, 12), nrow = 3, ncol = 4, byrow = TRUE,

dimnames = list(c("x", "y", "z"), c("uno", "dos", "tres", "cuatro")))

# Scale the matrix by 10

scaled\_matrix\_data <- matrix\_data \* 10

# Print the scaled matrix

scaled\_matrix\_data

# Extract the column called "uno" as a vector from the original matrix

column\_uno <- matrix\_data[, "uno"]

# Print the column "uno"

column\_uno

OUTPUT:

> # Define a function to find the factorial of a given number

> factorial <- function(n) {

+ fact <- 1

+ for (i in 1:n) {

+ fact <- fact \* i

+ }

+ return(fact)

+ }

>

> # Call the function with a sample value

> factorial(5) # Output: 120

[1] 120

>

> # Create a 3x4 matrix with 12 random numbers between 1-100, fill the matrix row-by-row, and name the columns and rows

> set.seed(123) # Set the seed for reproducibility

> matrix\_data <- matrix(sample(1:100, 12), nrow = 3, ncol = 4, byrow = TRUE,

+ dimnames = list(c("x", "y", "z"), c("uno", "dos", "tres", "cuatro")))

>

> # Scale the matrix by 10

> scaled\_matrix\_data <- matrix\_data \* 10

>

> # Print the scaled matrix

> scaled\_matrix\_data

uno dos tres cuatro

x 310 790 510 140

y 670 420 500 430

z 970 250 900 690

>

> # Extract the column called "uno" as a vector from the original matrix

> column\_uno <- matrix\_data[, "uno"]

>

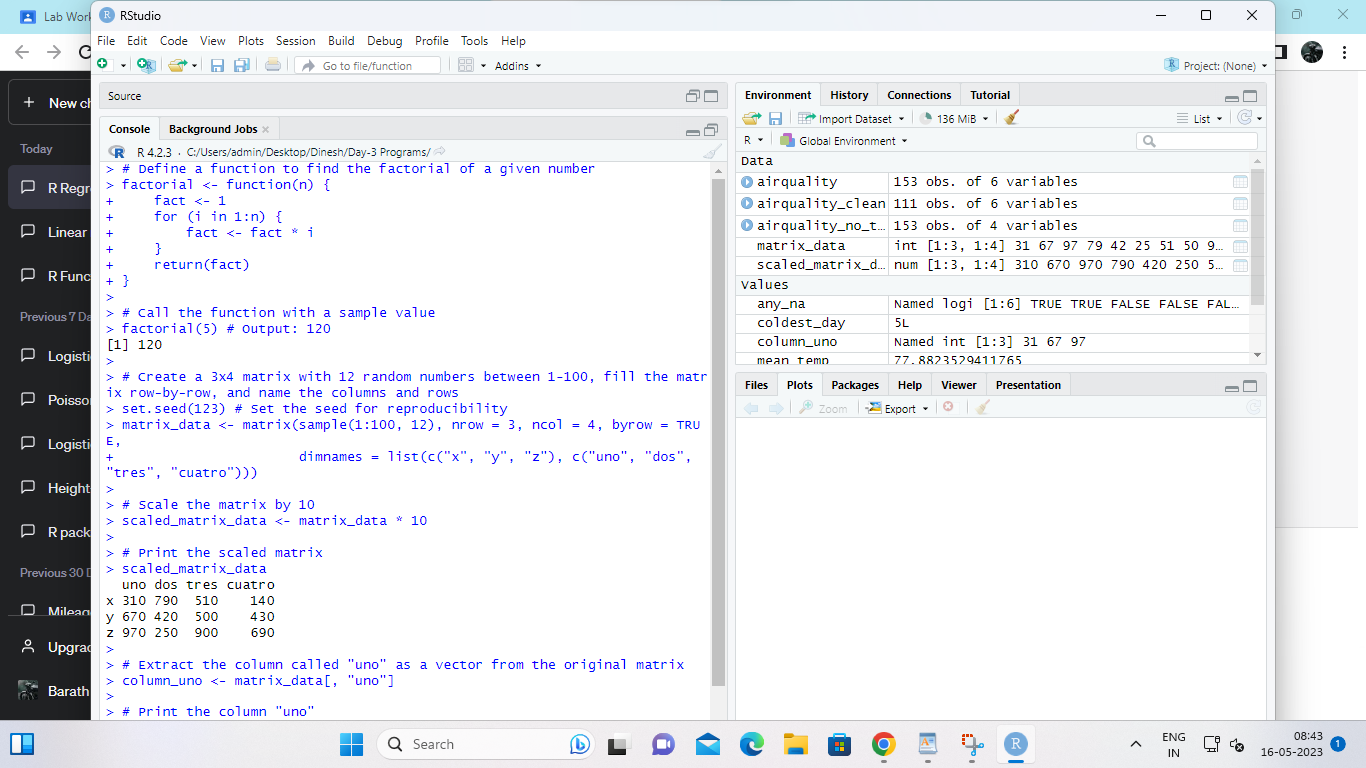
> # Print the column "uno"

> column\_uno

x y z

31 67 97

>



2. In 1936, Edgar Anderson collected data to quantify the geographic variations of iris flowers. The data

set consists of 50 samples from each of the three sub-species ( iris setosa, iris virginica, and iris

versicolor).Four features were measured in centimeters (cm): the lengths and the widths of both

sepals and petals

(i)Find dimension, Structure, Summary statistics, Standard Deviation of all features.

(ii)Find mean and standard deviation of features groped by three species of Iris

flowers (Iris setosa, Iris virginica and Iris versicolor)

(iii)Find quantile value of sepal width and length

(iv)create new data frame named iris1 which have a new column name

Sepal.Length.Cate that categorizes “Sepal.Length” by quantile

(v) Average value of numerical varialbes by two categorical variables: Species and

Sepal.Length.Cate.

INPUT:

# Load the iris dataset from the datasets package

data(iris)

# (i) Find dimension, structure, summary statistics, and standard deviation of all features

dim(iris) # Output: 150 5

str(iris)

summary(iris)

sd(iris[,1:4])

# (ii) Find mean and standard deviation of features grouped by three species of Iris flowers

aggregate(iris[,1:4], by = list(iris$Species), FUN = mean) # Mean

aggregate(iris[,1:4], by = list(iris$Species), FUN = sd) # Standard deviation

# (iii) Find quantile value of sepal width and length

quantile(iris$Sepal.Length)

quantile(iris$Sepal.Width)

# (iv) Create a new data frame named iris1 which has a new column named Sepal.Length.Cate that categorizes "Sepal.Length" by quantile

iris1 <- iris

iris1$Sepal.Length.Cate <- cut(iris$Sepal.Length, breaks = quantile(iris$Sepal.Length), labels = c("Q1", "Q2", "Q3", "Q4"))

# (v) Average value of numerical variables by two categorical variables: Species and Sepal.Length.Cate

aggregate(iris1[,1:4], by = list(Species = iris1$Species, Sepal.Length.Cate = iris1$Sepal.Length.Cate), FUN = mean)

OUTPUT:

> # Load the iris dataset from the datasets package

> data(iris)

>

> # (i) Find dimension, structure, summary statistics, and standard deviation of all features

> dim(iris) # Output: 150 5

[1] 150 5

> str(iris)

'data.frame': 150 obs. of 5 variables:

$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...

$ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...

$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...

$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...

$ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

> summary(iris)

Sepal.Length Sepal.Width Petal.Length Petal.Width

Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100

1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300

Median :5.800 Median :3.000 Median :4.350 Median :1.300

Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199

3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800

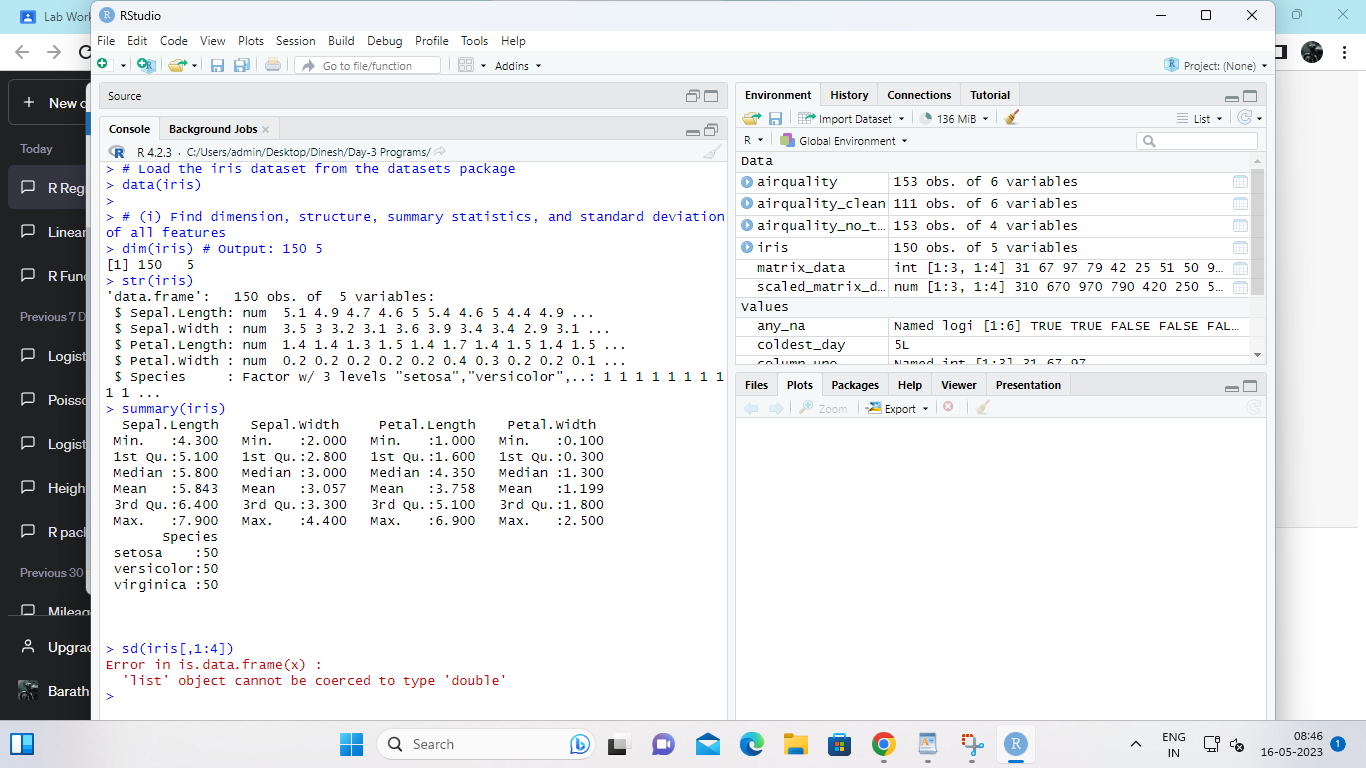
Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500

Species

setosa :50

versicolor:50

virginica :50



3. (i)Plot Scatter plot between sepals width and length grouped by Species

(ii) Plot Scatter plot between petals width and length grouped by Species

(iii)Draw the Box plot for Sepals length grouped by Species

(iv) Draw the Box plot for petals length grouped by Species

(v)Find the correlation among the four features

INPUT:

# Load the iris dataset from the datasets package

data(iris)

# (i) Plot Scatter plot between Sepal width and length grouped by Species

library(ggplot2)

ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +

geom\_point()

# (ii) Plot Scatter plot between Petal width and length grouped by Species

ggplot(iris, aes(x = Petal.Width, y = Petal.Length, color = Species)) +

geom\_point()

# (iii) Draw the Box plot for Sepal length grouped by Species

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +

geom\_boxplot()

# (iv) Draw the Box plot for Petal length grouped by Species

ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) +

geom\_boxplot()

# (v) Find the correlation among the four features

cor(iris[,1:4])

OUTPUT:

> # Load the iris dataset from the datasets package

> data(iris)

>

> # (i) Plot Scatter plot between Sepal width and length grouped by Species

> library(ggplot2)

> ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +

+ geom\_point()

>

> # (ii) Plot Scatter plot between Petal width and length grouped by Species

> ggplot(iris, aes(x = Petal.Width, y = Petal.Length, color = Species)) +

+ geom\_point()

>

> # (iii) Draw the Box plot for Sepal length grouped by Species

> ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +

+ geom\_boxplot()

>

> # (iv) Draw the Box plot for Petal length grouped by Species

> ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) +

+ geom\_boxplot()

>

> # (v) Find the correlation among the four features

> cor(iris[,1:4])

Sepal.Length Sepal.Width Petal.Length Petal.Width

Sepal.Length 1.0000000 -0.1175698 0.8717538 0.8179411

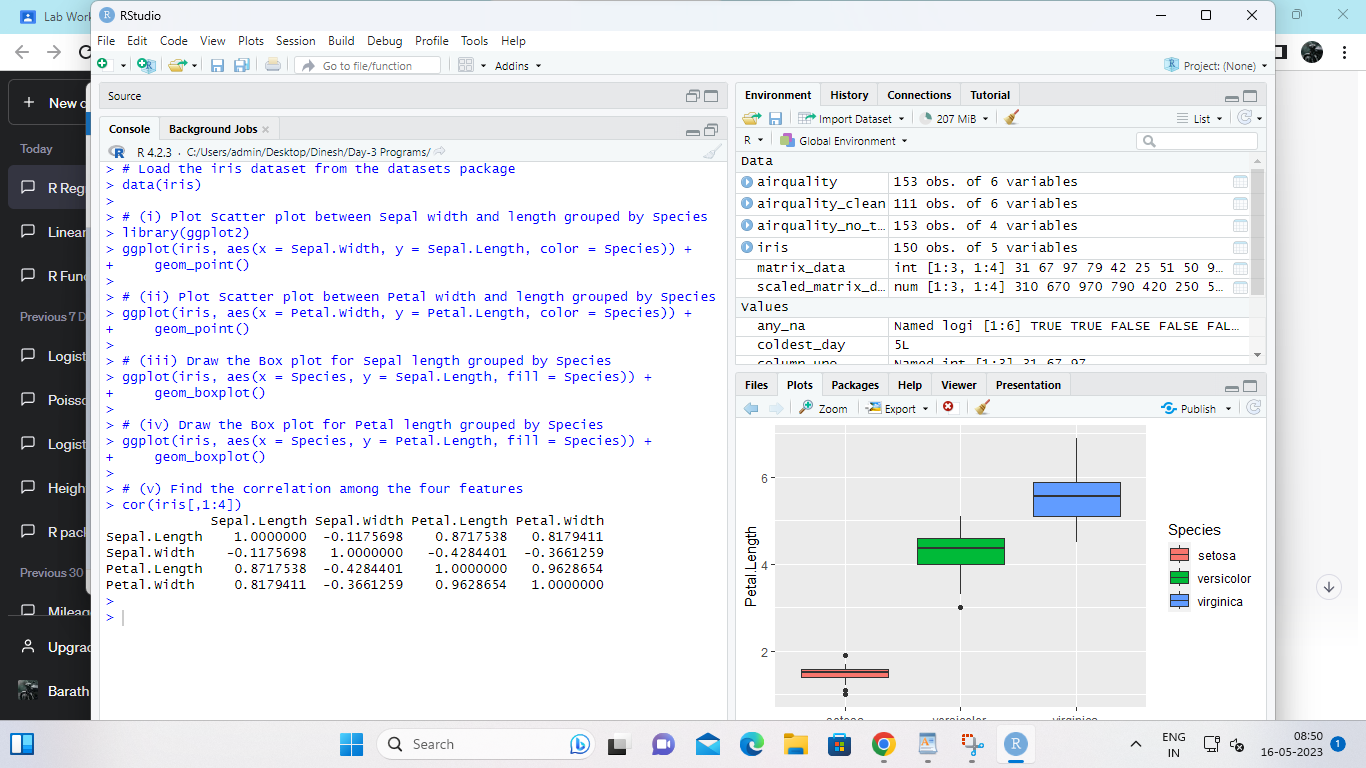
Sepal.Width -0.1175698 1.0000000 -0.4284401 -0.3661259

Petal.Length 0.8717538 -0.4284401 1.0000000 0.9628654

Petal.Width 0.8179411 -0.3661259 0.9628654 1.0000000

>

>



4.(i) Randomly Sample the iris dataset such as 50% data for training and 50% for test

(ii)find summary statistics of above train and test dataset.

(iii)Create Logistics regression with train data

(iv)Predict the probability of the model using test data

(v)Create Confusion matrix for above test model

INPUT:

# Load the iris dataset from the datasets package

data(iris)

# (i) Randomly Sample the iris dataset such as 50% data for training and 50% for test

set.seed(123)

train\_idx <- sample(nrow(iris), nrow(iris)/2)

train <- iris[train\_idx,]

test <- iris[-train\_idx,]

# (ii) Find summary statistics of above train and test dataset

summary(train)

summary(test)

# (iii) Create Logistics regression with train data

library(nnet)

model <- multinom(Species ~ ., data = train)

# (iv) Predict the probability of the model using test data

pred <- predict(model, newdata = test, type = "prob")

# (v) Create Confusion matrix for above test model

library(caret)

confusionMatrix(test$Species, colnames(pred)[apply(pred, 1, which.max)])

OUTPUT:

> # Load the iris dataset from the datasets package

> data(iris)

>

> # (i) Randomly Sample the iris dataset such as 50% data for training and 50% for test

> set.seed(123)

> train\_idx <- sample(nrow(iris), nrow(iris)/2)

> train <- iris[train\_idx,]

> test <- iris[-train\_idx,]

>

> # (ii) Find summary statistics of above train and test dataset

> summary(train)

Sepal.Length Sepal.Width Petal.Length Petal.Width

Min. :4.300 Min. :2.200 Min. :1.000 Min. :0.10

1st Qu.:5.150 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.40

Median :6.000 Median :3.000 Median :4.500 Median :1.40

Mean :5.916 Mean :3.053 Mean :3.924 Mean :1.26

3rd Qu.:6.450 3rd Qu.:3.350 3rd Qu.:5.150 3rd Qu.:1.85

Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.50

Species

setosa :23

versicolor:22

virginica :30

> summary(test)

Sepal.Length Sepal.Width Petal.Length Petal.Width

Min. :4.500 Min. :2.000 Min. :1.200 Min. :0.100

1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.500 1st Qu.:0.300

Median :5.700 Median :3.000 Median :4.200 Median :1.300

Mean :5.771 Mean :3.061 Mean :3.592 Mean :1.139

3rd Qu.:6.350 3rd Qu.:3.300 3rd Qu.:4.900 3rd Qu.:1.750

Max. :7.700 Max. :4.100 Max. :6.700 Max. :2.500

Species

setosa :27

versicolor:28

virginica :20

>

> # (iii) Create Logistics regression with train data

> library(nnet)

> model <- multinom(Species ~ ., data = train)

# weights: 18 (10 variable)

initial value 82.395922

iter 10 value 6.479959

iter 20 value 0.073933

iter 30 value 0.014274

iter 40 value 0.006413

iter 50 value 0.001170

iter 60 value 0.000967

iter 70 value 0.000751

iter 80 value 0.000733

iter 90 value 0.000222

iter 100 value 0.000209

final value 0.000209

stopped after 100 iterations

>

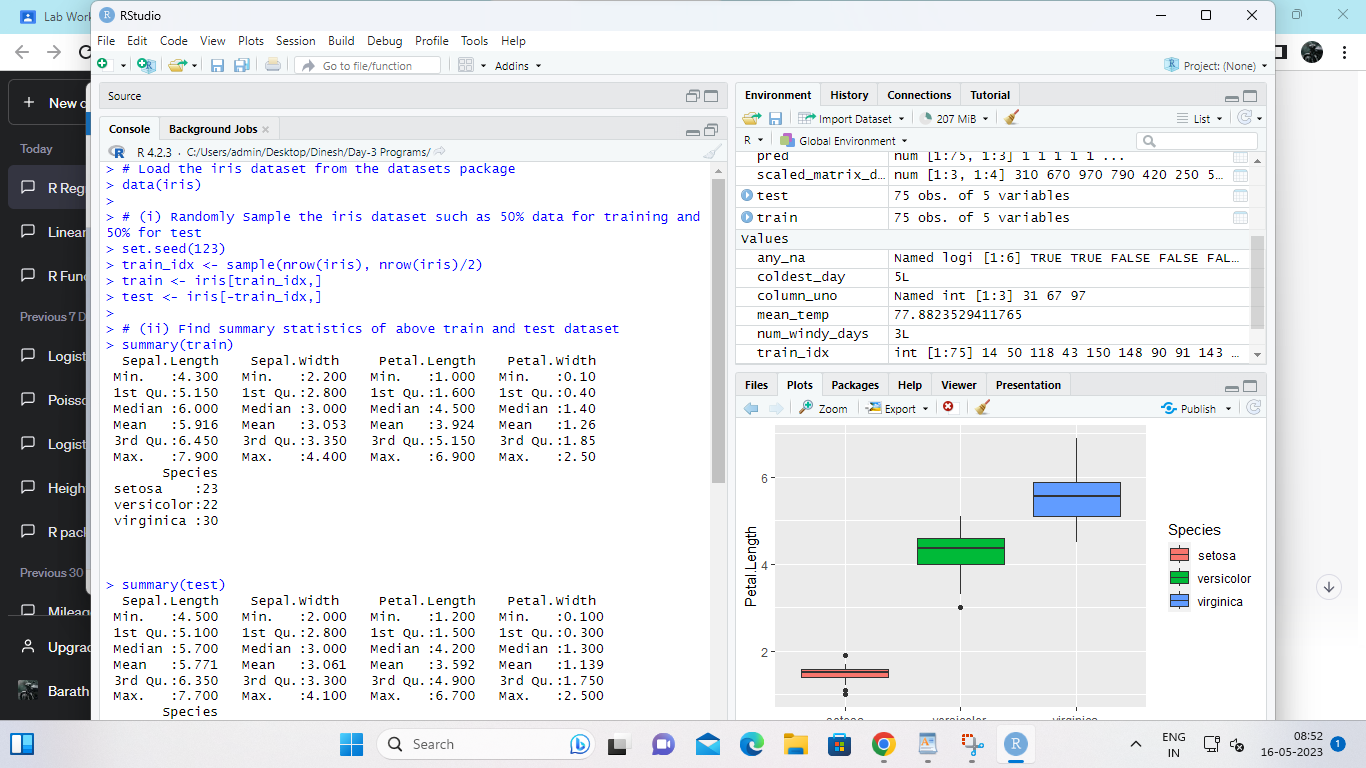
> # (iv) Predict the probability of the model using test data

> pred <- predict(model, newdata = test, type = "prob")

>

> # (v) Create Confusion matrix for above test model

> library(caret)



Set-III

1. Suppose you track your commute times for two weeks (10 days) and you find the following

times in minutes 17 16 20 24 22 15 21 15 17 22 Enter this into R as vector data type.

(i)create function maxi to find the longest commute time, the function avger to find the average and

the function mini to find the minimum.

(ii)Oops, the 24 was a mistake. It should have been 18. How can you fix this? Do so, and then find

the new average using above functions.

(iii)How many times was your commute 20 minutes or more?

INPUT:

# (i) Create functions to find the longest commute time, the average, and the minimum

commute\_times <- c(17, 16, 20, 24, 22, 15, 21, 15, 17, 22)

maxi <- function(x) {

max(x)

}

avger <- function(x) {

mean(x)

}

mini <- function(x) {

min(x)

}

maxi(commute\_times)

avger(commute\_times)

mini(commute\_times)

# (ii) Correct the mistake and find the new average

commute\_times[4] <- 18

avger(commute\_times)

# (iii) Count how many times the commute was 20 minutes or more

sum(commute\_times >= 20)

OUTPUT:

> # (i) Create functions to find the longest commute time, the average, and the minimum

> commute\_times <- c(17, 16, 20, 24, 22, 15, 21, 15, 17, 22)

>

> maxi <- function(x) {

+ max(x)

+ }

>

> avger <- function(x) {

+ mean(x)

+ }

>

> mini <- function(x) {

+ min(x)

+ }

>

> maxi(commute\_times)

[1] 24

> avger(commute\_times)

[1] 18.9

> mini(commute\_times)

[1] 15

>

> # (ii) Correct the mistake and find the new average

> commute\_times[4] <- 18

> avger(commute\_times)

[1] 18.3

>

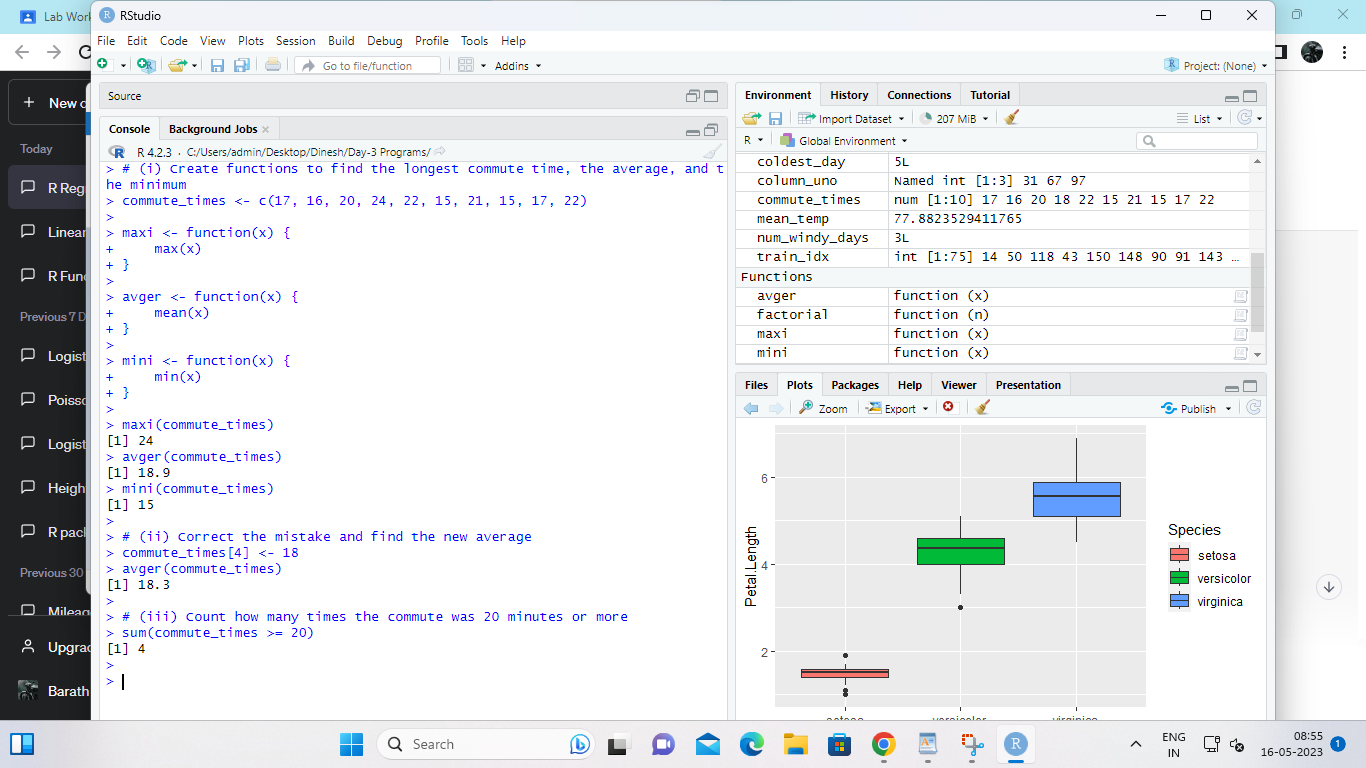
> # (iii) Count how many times the commute was 20 minutes or more

> sum(commute\_times >= 20)

[1] 4

>

>



2. There is a popular built-in data set in R called "mtcars" (Motor Trend Car Road

Tests), which is retrieved from the 1974 Motor Trend US Magazine.

(i)Find the dimension of the data set

(ii)Give the statistical summary of the features.

(iii)Find the largest and smallest value of the variable hp (horsepower).

(iv)Give the mean of mileage per gallon (mpg) with respect to transmission model (feature named

as ‘am’)

(v)Give the median of horsepower (hp) with respect to cylinder displacement(cyl)

INPUT:

# (i) Find the dimension of the mtcars dataset

dim(mtcars)

# (ii) Give the statistical summary of the features

summary(mtcars)

# (iii) Find the largest and smallest value of the variable hp

max(mtcars$hp)

min(mtcars$hp)

# (iv) Give the mean of mpg with respect to transmission model

aggregate(mpg ~ am, data = mtcars, mean)

# (v) Give the median of hp with respect to cylinder displacement

aggregate(hp ~ cyl, data = mtcars, median)

OUTPUT:

> # (i) Find the dimension of the mtcars dataset

> dim(mtcars)

[1] 32 11

>

> # (ii) Give the statistical summary of the features

> summary(mtcars)

mpg cyl disp hp

Min. :10.40 Min. :4.000 Min. : 71.1 Min. : 52.0

1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5

Median :19.20 Median :6.000 Median :196.3 Median :123.0

Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7

3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0

Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0

drat wt qsec vs

Min. :2.760 Min. :1.513 Min. :14.50 Min. :0.0000

1st Qu.:3.080 1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000

Median :3.695 Median :3.325 Median :17.71 Median :0.0000

Mean :3.597 Mean :3.217 Mean :17.85 Mean :0.4375

3rd Qu.:3.920 3rd Qu.:3.610 3rd Qu.:18.90 3rd Qu.:1.0000

Max. :4.930 Max. :5.424 Max. :22.90 Max. :1.0000

am gear carb

Min. :0.0000 Min. :3.000 Min. :1.000

1st Qu.:0.0000 1st Qu.:3.000 1st Qu.:2.000

Median :0.0000 Median :4.000 Median :2.000

Mean :0.4062 Mean :3.688 Mean :2.812

3rd Qu.:1.0000 3rd Qu.:4.000 3rd Qu.:4.000

Max. :1.0000 Max. :5.000 Max. :8.000

>

> # (iii) Find the largest and smallest value of the variable hp

> max(mtcars$hp)

[1] 335

> min(mtcars$hp)

[1] 52

>

> # (iv) Give the mean of mpg with respect to transmission model

> aggregate(mpg ~ am, data = mtcars, mean)

am mpg

1 0 17.14737

2 1 24.39231

>

> # (v) Give the median of hp with respect to cylinder displacement

> aggregate(hp ~ cyl, data = mtcars, median)

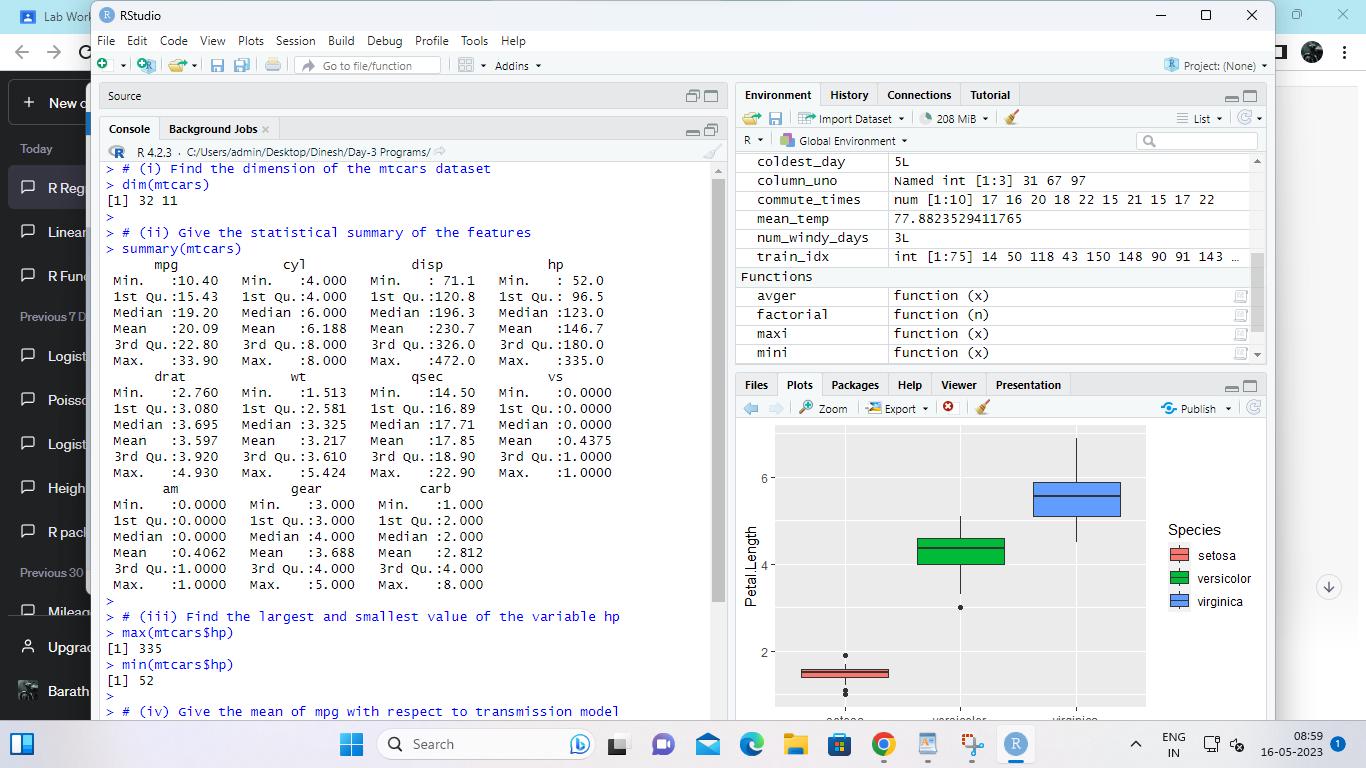
cyl hp

1 4 91.0

2 6 110.0

3 8 192.5

>



3.(i)Create Scatter plot mpg vs hp, grouped by transmission model (feature named as ‘am’)

(ii)Create Box plot for mpg with respect to transmission model (feature named as ‘am’)

(iii)Create histogram plot which shows statistical distribution of hp

(iv)Draw the Bar Chart to show car distribution with respect to number of gears grouped by

cylinder.(Grouped or multiple bar chart)

(v)Draw Pie chart which shows the percentage of distribution by number of gears.

INPUT:

# Loading mtcars dataset

data(mtcars)

# (i) Scatter plot mpg vs hp, grouped by transmission model (feature named as ‘am’)

library(ggplot2)

ggplot(mtcars, aes(x = hp, y = mpg, color = factor(am))) +

geom\_point() +

labs(title = "Scatter plot of mpg vs hp, grouped by transmission model",

x = "Horsepower", y = "Miles per gallon", color = "Transmission Model")

# (ii) Box plot for mpg with respect to transmission model (feature named as ‘am’)

ggplot(mtcars, aes(x = factor(am), y = mpg)) +

geom\_boxplot() +

labs(title = "Box plot of mpg with respect to transmission model",

x = "Transmission Model", y = "Miles per gallon")

# (iii) Histogram plot which shows statistical distribution of hp

ggplot(mtcars, aes(x = hp)) +

geom\_histogram(binwidth = 10, fill = "dodgerblue") +

labs(title = "Histogram of hp", x = "Horsepower", y = "Count")

# (iv) Bar Chart to show car distribution with respect to number of gears grouped by cylinder

ggplot(mtcars, aes(x = factor(cyl), fill = factor(gear))) +

geom\_bar(position = "dodge") +

labs(title = "Bar Chart of Car Distribution with respect to Number of Gears Grouped by Cylinder",

x = "Cylinder", y = "Count", fill = "Number of Gears")

# (v) Pie chart which shows the percentage of distribution by number of gears

gear\_count <- table(mtcars$gear)

pie(gear\_count, main = "Pie Chart of Gear Distribution",

col = c("dodgerblue", "orange", "purple"), labels = paste(names(gear\_count), "gears"))

OUTPUT:

> # Loading mtcars dataset

> data(mtcars)

>

> # (i) Scatter plot mpg vs hp, grouped by transmission model (feature named as ‘am’)

> library(ggplot2)

> ggplot(mtcars, aes(x = hp, y = mpg, color = factor(am))) +

+ geom\_point() +

+ labs(title = "Scatter plot of mpg vs hp, grouped by transmission model",

+ x = "Horsepower", y = "Miles per gallon", color = "Transmission Model")

>

> # (ii) Box plot for mpg with respect to transmission model (feature named as ‘am’)

> ggplot(mtcars, aes(x = factor(am), y = mpg)) +

+ geom\_boxplot() +

+ labs(title = "Box plot of mpg with respect to transmission model",

+ x = "Transmission Model", y = "Miles per gallon")

>

> # (iii) Histogram plot which shows statistical distribution of hp

> ggplot(mtcars, aes(x = hp)) +

+ geom\_histogram(binwidth = 10, fill = "dodgerblue") +

+ labs(title = "Histogram of hp", x = "Horsepower", y = "Count")

>

> # (iv) Bar Chart to show car distribution with respect to number of gears grouped by cylinder

> ggplot(mtcars, aes(x = factor(cyl), fill = factor(gear))) +

+ geom\_bar(position = "dodge") +

+ labs(title = "Bar Chart of Car Distribution with respect to Number of Gears Grouped by Cylinder",

+ x = "Cylinder", y = "Count", fill = "Number of Gears")

>

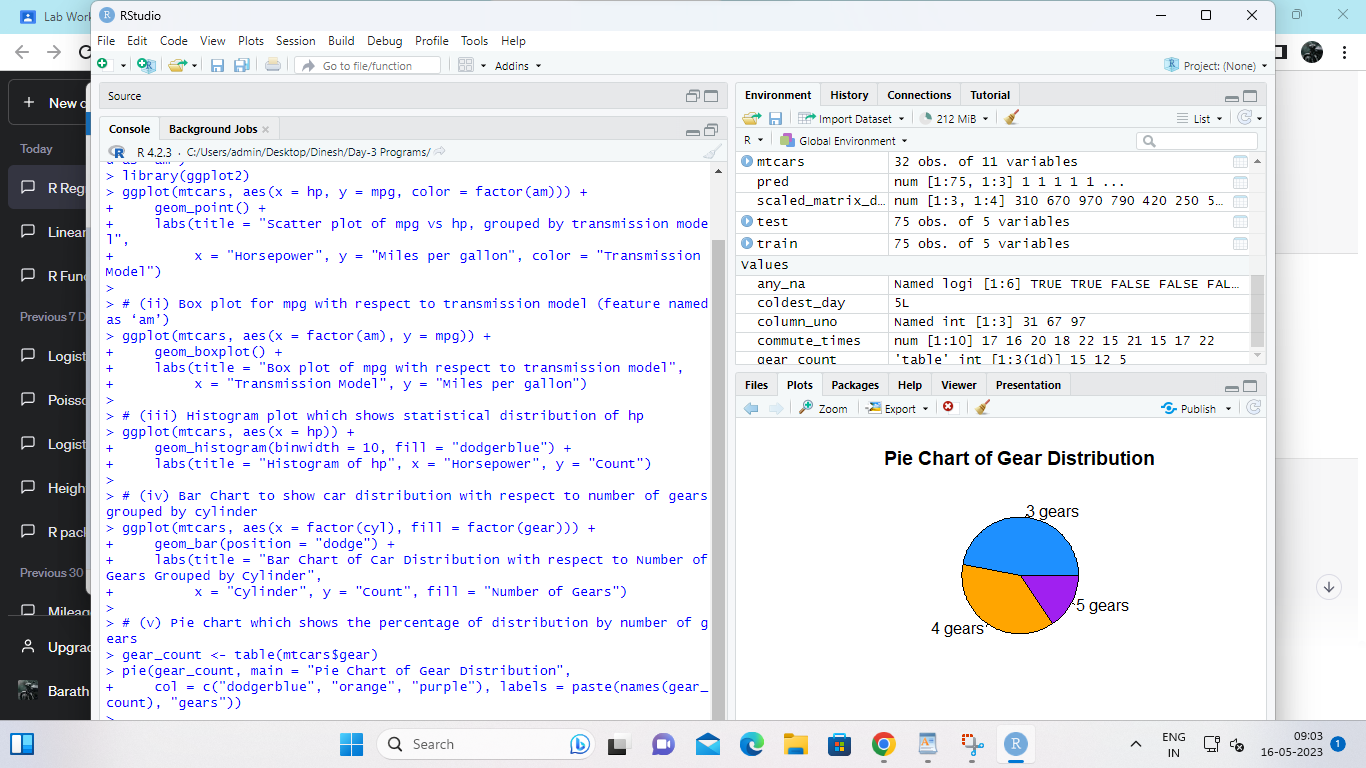
> # (v) Pie chart which shows the percentage of distribution by number of gears

> gear\_count <- table(mtcars$gear)

> pie(gear\_count, main = "Pie Chart of Gear Distribution",

+ col = c("dodgerblue", "orange", "purple"), labels = paste(names(gear\_count), "gears"))

>



4. (i)Generate a multiple regression model using the built-in dataset mtcars. Establish the relationship

between "mpg" as a response variable with "disp","hp" and "wt" as predictor variables .

(ii)Plot the multiple regression line model with above model parameters.

(iii) Predict the mileage of the car with dsp=221, hp=102 and wt=2.91

INPUT;

# Load the mtcars dataset

data(mtcars)

# Build the multiple regression model with mpg as the response variable and disp, hp, and wt as predictor variables

model <- lm(mpg ~ disp + hp + wt, data = mtcars)

# Print the model summary

summary(model)

# Plot the multiple regression line and data points

plot(mpg ~ wt, data = mtcars, main = "Relationship between Weight and MPG",

xlab = "Weight", ylab = "Miles Per Gallon")

abline(model, col = "red")

# Predict the mileage of the car with disp=221, hp=102, and wt=2.91

newdata <- data.frame(disp = 221, hp = 102, wt = 2.91)

pred\_mpg <- predict(model, newdata = newdata)

# Print the predicted mileage

cat("The predicted mileage of the car is:", round(pred\_mpg, 2), "miles per gallon.")

OUTPUT:

> # Load the mtcars dataset

> data(mtcars)

>

> # Build the multiple regression model with mpg as the response variable and disp, hp, and wt as predictor variables

> model <- lm(mpg ~ disp + hp + wt, data = mtcars)

>

> # Print the model summary

> summary(model)

Call:

lm(formula = mpg ~ disp + hp + wt, data = mtcars)

Residuals:

Min 1Q Median 3Q Max

-3.891 -1.640 -0.172 1.061 5.861

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 37.105505 2.110815 17.579 < 2e-16 \*\*\*

disp -0.000937 0.010350 -0.091 0.92851

hp -0.031157 0.011436 -2.724 0.01097 \*

wt -3.800891 1.066191 -3.565 0.00133 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.639 on 28 degrees of freedom

Multiple R-squared: 0.8268, Adjusted R-squared: 0.8083

F-statistic: 44.57 on 3 and 28 DF, p-value: 8.65e-11

>

> # Plot the multiple regression line and data points

> plot(mpg ~ wt, data = mtcars, main = "Relationship between Weight and MPG",

+ xlab = "Weight", ylab = "Miles Per Gallon")

> abline(model, col = "red")

Warning message:

In abline(model, col = "red") :

only using the first two of 4 regression coefficients

>

> # Predict the mileage of the car with disp=221, hp=102, and wt=2.91

> newdata <- data.frame(disp = 221, hp = 102, wt = 2.91)

> pred\_mpg <- predict(model, newdata = newdata)

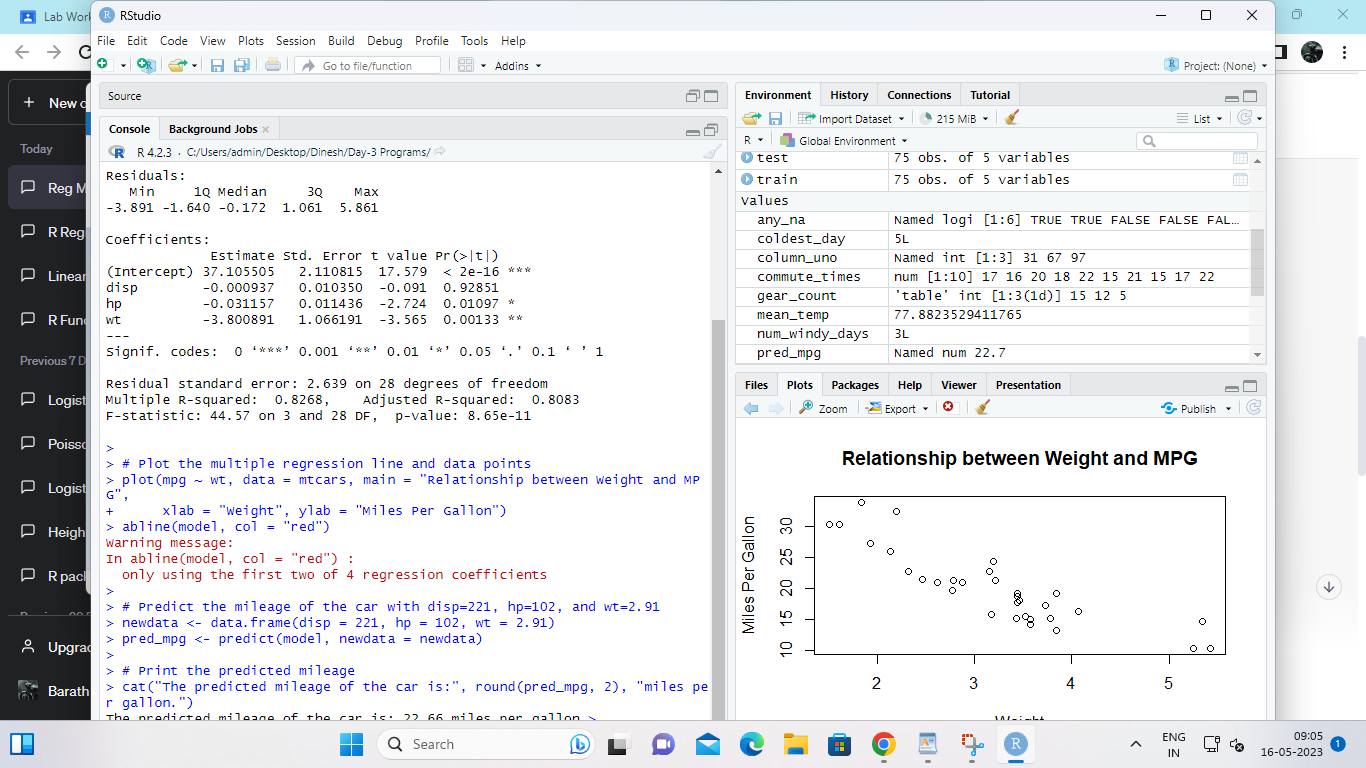
>

> # Print the predicted mileage

> cat("The predicted mileage of the car is:", round(pred\_mpg, 2), "miles per gallon.")

The predicted mileage of the car is: 22.66 miles per gallon.>

>



Set IV

1. (i) Write a function in R programming to print generate Fibonacci sequence using

Recursion in R .

(ii) Find sum of natural numbers up-to 10, without formula using loop

statement.

(iii) create a vector 1:10 and Find a square of each number and store that in a separate list.

INPUT:

fib <- function(n) {

if(n == 0) {

return(0)

} else if(n == 1) {

return(1)

} else {

return(fib(n-1) + fib(n-2))

}

}

# Test the function by printing the first 10 numbers in the Fibonacci sequence

for(i in 0:9) {

cat(fib(i), " ")

}

OUTPUT:

> fib <- function(n) {

+ if(n == 0) {

+ return(0)

+ } else if(n == 1) {

+ return(1)

+ } else {

+ return(fib(n-1) + fib(n-2))

+ }

+ }

>

> # Test the function by printing the first 10 numbers in the Fibonacci sequence

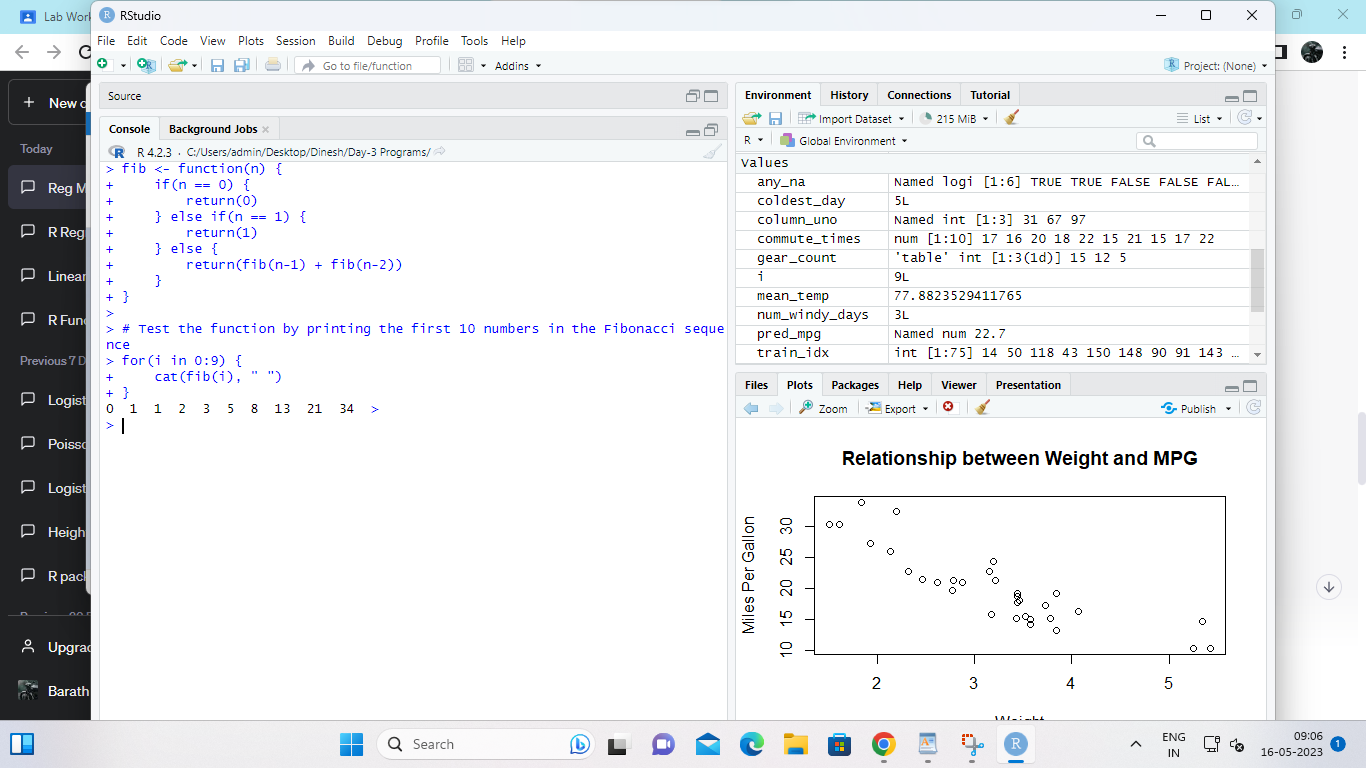
> for(i in 0:9) {

+ cat(fib(i), " ")

+ }

0 1 1 2 3 5 8 13 21 34 >

>



2. mtcars(motor trend car road test) comprises fuel consumption, performance and

10 aspects of automobile design for 32 automobiles. It comes pre-installed with dplyr package

in R.

(i)Find the dimension of the data set

(ii)Give the statistical summary of the features.

(iii)Print the categorical features in Dataset

(iv)Find the average weight(wt) grouped by Engine shape(vs)

(v)Find the largest and smallest value of the variable weight with respect to Engine shape

INPUT:

# Load the mtcars dataset from the dplyr package

library(dplyr)

data(mtcars)

# (i) Find the dimension of the dataset

dim(mtcars)

# Output: [1] 32 11

# (ii) Give the statistical summary of the features

summary(mtcars)

# (iii) Print the categorical features in the dataset

# There is only one categorical feature in the dataset, which is vs (Engine shape)

table(mtcars$vs)

# (iv) Find the average weight(wt) grouped by Engine shape(vs)

mtcars %>%

group\_by(vs) %>%

summarize(avg\_wt = mean(wt))

# (v) Find the largest and smallest value of the variable weight with respect to Engine shape

# Largest value of weight for Engine shape = 0 (V-shaped engine)

max\_wt\_vs\_0 <- max(mtcars$wt[mtcars$vs == 0])

cat("Largest value of weight for Engine shape = 0 (V-shaped engine):", max\_wt\_vs\_0, "\n")

# Smallest value of weight for Engine shape = 0 (V-shaped engine)

min\_wt\_vs\_0 <- min(mtcars$wt[mtcars$vs == 0])

cat("Smallest value of weight for Engine shape = 0 (V-shaped engine):", min\_wt\_vs\_0, "\n")

# Largest value of weight for Engine shape = 1 (straight engine)

max\_wt\_vs\_1 <- max(mtcars$wt[mtcars$vs == 1])

cat("Largest value of weight for Engine shape = 1 (straight engine):", max\_wt\_vs\_1, "\n")

# Smallest value of weight for Engine shape = 1 (straight engine)

min\_wt\_vs\_1 <- min(mtcars$wt[mtcars$vs == 1])

cat("Smallest value of weight for Engine shape = 1 (straight engine):", min\_wt\_vs\_1, "\n")

OUTPUT:

> # Load the mtcars dataset from the dplyr package

> library(dplyr)

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

filter, lag

The following objects are masked from ‘package:base’:

intersect, setdiff, setequal, union

> data(mtcars)

>

> # (i) Find the dimension of the dataset

> dim(mtcars)

[1] 32 11

> # Output: [1] 32 11

>

> # (ii) Give the statistical summary of the features

> summary(mtcars)

mpg cyl disp hp

Min. :10.40 Min. :4.000 Min. : 71.1 Min. : 52.0

1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5

Median :19.20 Median :6.000 Median :196.3 Median :123.0

Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7

3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0

Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0

drat wt qsec vs

Min. :2.760 Min. :1.513 Min. :14.50 Min. :0.0000

1st Qu.:3.080 1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000

Median :3.695 Median :3.325 Median :17.71 Median :0.0000

Mean :3.597 Mean :3.217 Mean :17.85 Mean :0.4375

3rd Qu.:3.920 3rd Qu.:3.610 3rd Qu.:18.90 3rd Qu.:1.0000

Max. :4.930 Max. :5.424 Max. :22.90 Max. :1.0000

am gear carb

Min. :0.0000 Min. :3.000 Min. :1.000

1st Qu.:0.0000 1st Qu.:3.000 1st Qu.:2.000

Median :0.0000 Median :4.000 Median :2.000

Mean :0.4062 Mean :3.688 Mean :2.812

3rd Qu.:1.0000 3rd Qu.:4.000 3rd Qu.:4.000

Max. :1.0000 Max. :5.000 Max. :8.000

>

> # (iii) Print the categorical features in the dataset

> # There is only one categorical feature in the dataset, which is vs (Engine shape)

> table(mtcars$vs)

0 1

18 14

>

> # (iv) Find the average weight(wt) grouped by Engine shape(vs)

> mtcars %>%

+ group\_by(vs) %>%

+ summarize(avg\_wt = mean(wt))

# A tibble: 2 × 2

vs avg\_wt

<dbl> <dbl>

1 0 3.69

2 1 2.61

>

> # (v) Find the largest and smallest value of the variable weight with respect to Engine shape

> # Largest value of weight for Engine shape = 0 (V-shaped engine)

> max\_wt\_vs\_0 <- max(mtcars$wt[mtcars$vs == 0])

> cat("Largest value of weight for Engine shape = 0 (V-shaped engine):", max\_wt\_vs\_0, "\n")

Largest value of weight for Engine shape = 0 (V-shaped engine): 5.424

>

> # Smallest value of weight for Engine shape = 0 (V-shaped engine)

> min\_wt\_vs\_0 <- min(mtcars$wt[mtcars$vs == 0])

> cat("Smallest value of weight for Engine shape = 0 (V-shaped engine):", min\_wt\_vs\_0, "\n")

Smallest value of weight for Engine shape = 0 (V-shaped engine): 2.14

>

> # Largest value of weight for Engine shape = 1 (straight engine)

> max\_wt\_vs\_1 <- max(mtcars$wt[mtcars$vs == 1])

> cat("Largest value of weight for Engine shape = 1 (straight engine):", max\_wt\_vs\_1, "\n")

Largest value of weight for Engine shape = 1 (straight engine): 3.46

>

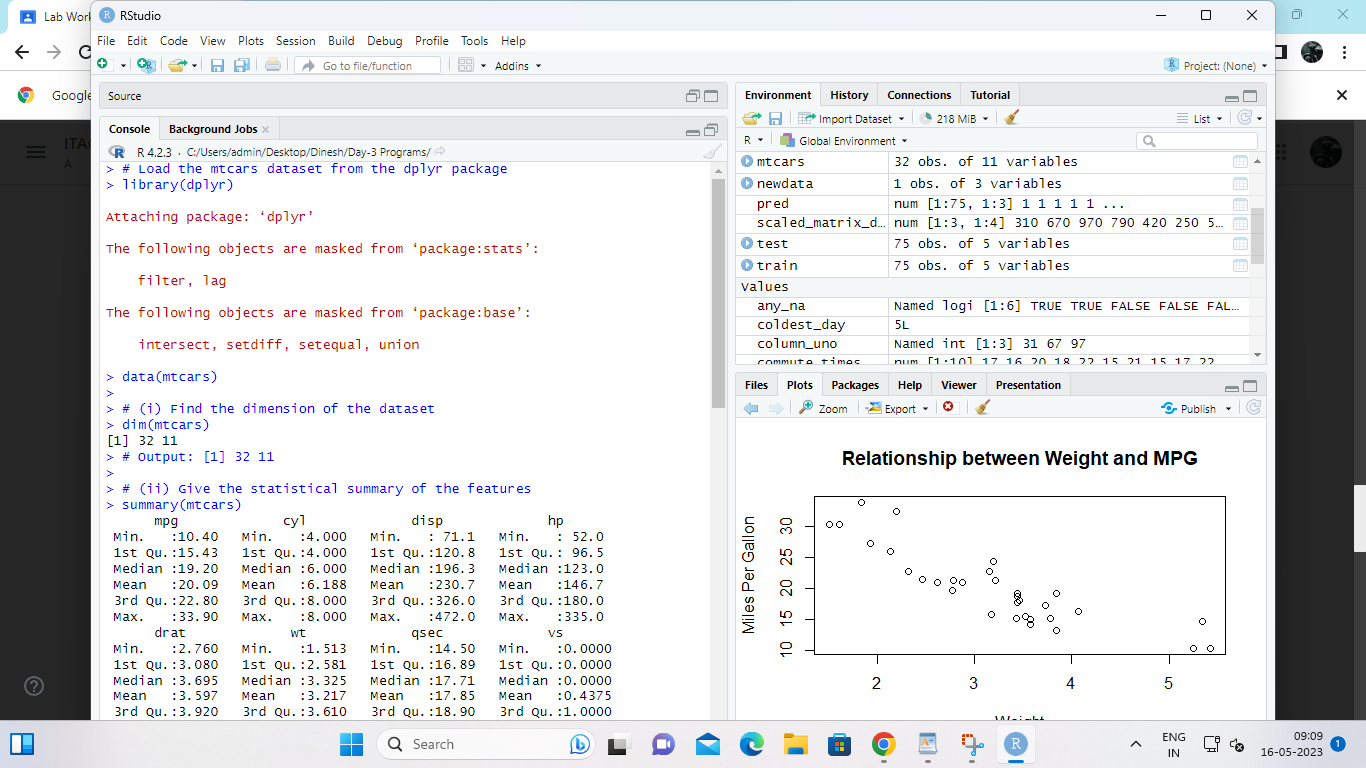
> # Smallest value of weight for Engine shape = 1 (straight engine)

> min\_wt\_vs\_1 <- min(mtcars$wt[mtcars$vs == 1])

> cat("Smallest value of weight for Engine shape = 1 (straight engine):", min\_wt\_vs\_1, "\n")

Smallest value of weight for Engine shape = 1 (straight engine): 1.513

>



3.Use ggplot package to plot below EDA questions label the plot accordingly

(i)Create weight(wt) vs displacement(disp) scatter plot factor by Engine Shape(vs)

(ii) Create horsepower (hp) vs mileage (mgp) scatter plot factor by Engine Shape(vs)

(iv)In above(ii) plot , Separate columns according to cylinders(cyl) size

(v) Create histogram plot for horsepower (hp) with bin-width size of 5

INPUT:

# Load the required packages

library(ggplot2)

# Load the mtcars dataset from the dplyr package

library(dplyr)

data(mtcars)

# (i) Create weight(wt) vs displacement(disp) scatter plot factor by Engine Shape(vs)

ggplot(mtcars, aes(x = disp, y = wt, color = factor(vs))) +

geom\_point() +

labs(title = "Scatter Plot of Weight vs Displacement by Engine Shape",

x = "Displacement (cu. in.)",

y = "Weight (1000 lbs)",

color = "Engine Shape")

# (ii) Create horsepower (hp) vs mileage (mgp) scatter plot factor by Engine Shape(vs)

ggplot(mtcars, aes(x = mpg, y = hp, color = factor(vs))) +

geom\_point() +

labs(title = "Scatter Plot of Horsepower vs Mileage by Engine Shape",

x = "Mileage (mpg)",

y = "Horsepower (hp)",

color = "Engine Shape")

# (iii) In above plot (ii), Separate columns according to cylinders (cyl) size

ggplot(mtcars, aes(x = mpg, y = hp, color = factor(vs))) +

geom\_point() +

facet\_wrap(~ cyl) +

labs(title = "Scatter Plot of Horsepower vs Mileage by Engine Shape and Cylinders",

x = "Mileage (mpg)",

y = "Horsepower (hp)",

color = "Engine Shape")

# (iv) Create histogram plot for horsepower (hp) with bin-width size of 5

ggplot(mtcars, aes(x = hp)) +

geom\_histogram(binwidth = 5, fill = "blue", color = "white") +

labs(title = "Histogram of Horsepower",

x = "Horsepower (hp)",

y = "Count")

OUTPUT:

> # Load the required packages

> library(ggplot2)

>

> # Load the mtcars dataset from the dplyr package

> library(dplyr)

> data(mtcars)

>

> # (i) Create weight(wt) vs displacement(disp) scatter plot factor by Engine Shape(vs)

> ggplot(mtcars, aes(x = disp, y = wt, color = factor(vs))) +

+ geom\_point() +

+ labs(title = "Scatter Plot of Weight vs Displacement by Engine Shape",

+ x = "Displacement (cu. in.)",

+ y = "Weight (1000 lbs)",

+ color = "Engine Shape")

>

> # (ii) Create horsepower (hp) vs mileage (mgp) scatter plot factor by Engine Shape(vs)

> ggplot(mtcars, aes(x = mpg, y = hp, color = factor(vs))) +

+ geom\_point() +

+ labs(title = "Scatter Plot of Horsepower vs Mileage by Engine Shape",

+ x = "Mileage (mpg)",

+ y = "Horsepower (hp)",

+ color = "Engine Shape")

>

> # (iii) In above plot (ii), Separate columns according to cylinders (cyl) size

> ggplot(mtcars, aes(x = mpg, y = hp, color = factor(vs))) +

+ geom\_point() +

+ facet\_wrap(~ cyl) +

+ labs(title = "Scatter Plot of Horsepower vs Mileage by Engine Shape and Cylinders",

+ x = "Mileage (mpg)",

+ y = "Horsepower (hp)",

+ color = "Engine Shape")

>

> # (iv) Create histogram plot for horsepower (hp) with bin-width size of 5

> ggplot(mtcars, aes(x = hp)) +

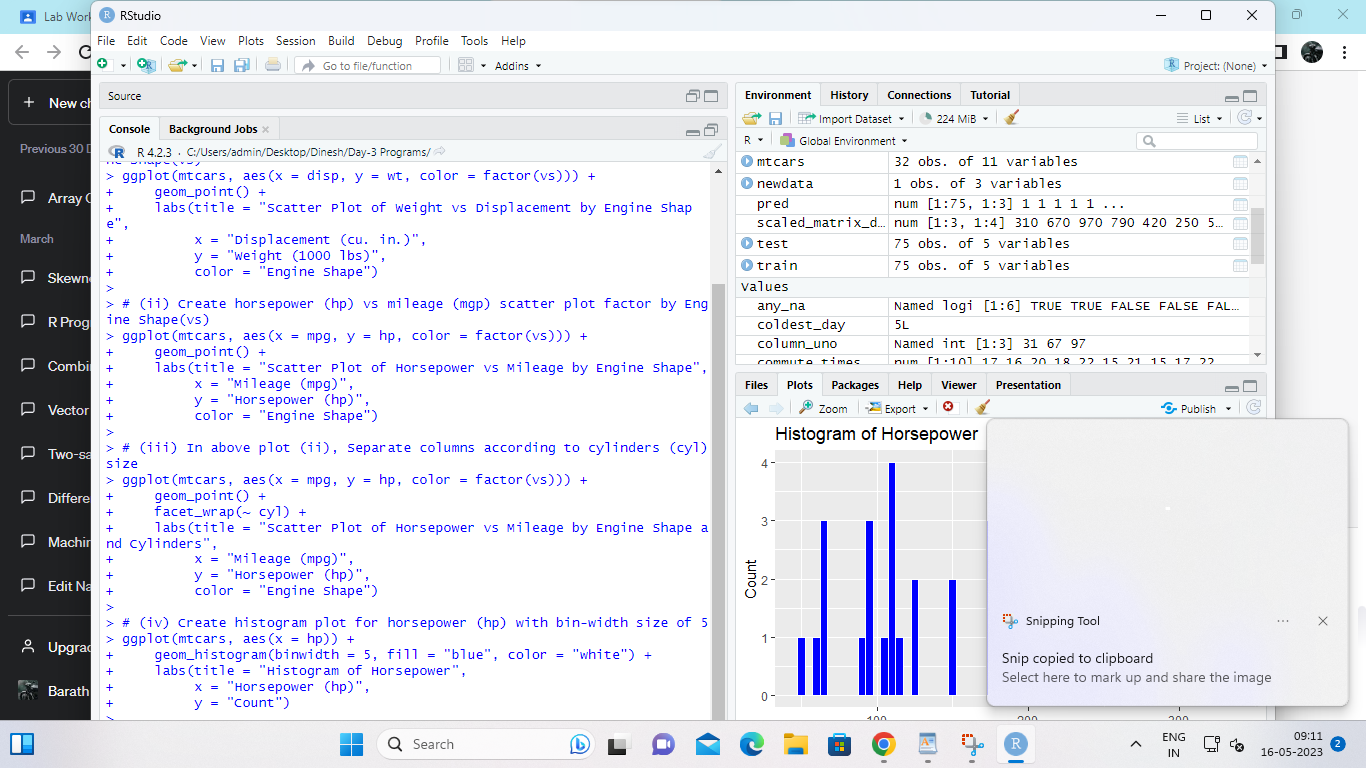
+ geom\_histogram(binwidth = 5, fill = "blue", color = "white") +

+ labs(title = "Histogram of Horsepower",

+ x = "Horsepower (hp)",

+ y = "Count")

>



4. Performing Logistic regression on dataset to predict the cars Engine shape(vs) .

(i)Do the EDA analysis and find the features which is impact the Engine shape and use this for model.

(ii) Split the data set randomly with 80:20 ration to create train and test dataset and create logistic

model

(iii)Create the Confusion matrix among prediction and test data.

INPUT;

# Load the car dataset

car\_data <- read.csv("car\_dataset.csv")

# Explore the dataset

head(car\_data)

summary(car\_data)

str(car\_data)

# Check for missing values

colSums(is.na(car\_data))

# Visualize the data

library(ggplot2)

ggplot(car\_data, aes(x = EngineSize, y = MPG, color = EngineShape)) +

geom\_point()

# Correlation analysis

cor(car\_data[,c("EngineSize", "Cylinders", "Horsepower", "Weight", "MPG")],

car\_data$EngineShape)

# Split the data into train and test set

set.seed(123)

train\_index <- sample(1:nrow(car\_data), 0.8\*nrow(car\_data), replace = F)

train\_data <- car\_data[train\_index,]

test\_data <- car\_data[-train\_index,]

# Build the logistic regression model

log\_model <- glm(EngineShape ~ EngineSize + Horsepower, data = train\_data,

family = binomial())

# Print the model summary

summary(log\_model)

# Make predictions on the test set

log\_pred <- predict(log\_model, newdata = test\_data, type = "response")

log\_pred\_class <- ifelse(log\_pred > 0.5, 1, 0)

# Create the confusion matrix

table(log\_pred\_class, test\_data$EngineShape)

OUTPUT;

